Yocto 1.2 M1 Fullpass Test

Test Report

Project: yocto

Author: admin

Printed by TestLink on 09/12/2011

1 Test Suite : Yocto 1.2 M1 Fullpass Test

1.1 Test Suite : System & Core OS

Test Case TC-1420: zypper command installed and workable	
Summary: check if zypper is installed and can work	
Steps: 1. Run command "zypper", and check the output	
Expected Results: Command "zypper" print the list of available global options and commands	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1421: zypper help search	
Summary: check help option with zypper command	
Steps: 1. Run "zypper help search" and check the output	
Expected Results: The command should print help for the search command	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1422: zypper search package	
Summary: search package with zypper	
Steps: 1. Run "zypper search package_name" and check the output, for example "zypper search avahi"	
Expected Results: The command should search package "avahi" is installed or not	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1423: zypper remove package
Summary:

remove package with zypper
Steps: 1. Run "zypper rm pakcage_name" and check the output, for example "zypper rm avahi"
Expected Results: The command should remove package "avahi"

Last Result	**Not Run**
Keywords:	None

Test Case TC-1424: zypper install package
Summary: install package with zypper
Steps: 1. Set up a yum based repository on local server 2. Build out a package, which does not need any run-time dependency package, with local poky tree. For example, package "man" 3. In target system, run "zypper addrepo http://ip_address_of_repository zypper_test_repo" 4. Run "zypper refresh" to refresh the zypper repository cache 5. Run "zypper install package_name" and check the output, for example "zypper install man" to install package, which has no run-time dependency
Expected Results: The command should install package "man"

Last Result	**Not Run**
Keywords:	None

Test Case TC-1425: zypper install dependency package
Summary: install dependency package with zypper
Steps: 1. Set up a yum based repository on local server 2. Build out a package, which does not need any run-time dependency package, with local poky tree. For example, package "mc" 3. In target system, run "zypper addrepo http://ip_address_of_repository zypper_test_repo" 4. Run "zypper refresh" to refresh the zypper repository cache 5. Run "zypper install package_name" and check the output, for example "zypper install mc" to install package, which needs run-time dependency packages installed also, like ncurses-terminfo.
Expected Results:

The command should install package "mc" and denpendency package ncurses-terminfo.
Last Result **Not Run**
Keywords: None

Test Case TC-1426: zypper install .all packages
Summary: install packages from all folder with zypper
Steps: 1. Set up a yum based repository on local server 2. Build out a package, which belongs to all folder, for example, xcursor-transparent-theme-dbg-0.1.1-r3.all.rpm. 3. In target system, run "zypper addrepo http://ip_address_of_repository zypper_test_repo" 4. Run "zypper refresh" to refresh the zypper repository cache 5. Run "zypper install xcursor-transparent-theme-dbg" and check the output
Expected Results: package install from all folder should be installed successfully with zypper
Last Result **Not Run**
Keywords: None

Test Case TC-1427: rpm query package
Summary: make sure rootfs image is built with rpm packages
Steps: 1. launch terminal 2. run command "rpm -qa", which lists all existing packages in system
Expected Results: "rpm -qa" should print all existing packages in system
Last Result **Not Run**
Keywords: None

Test Case TC-1428: rpm install package
Summary: rpm format package can be installed
Steps: 1. Get a RPM package(for example, avahi or powertop) from zypper repository or build one on local machine 2. Copy the package into image, run command "rpm -ivh package_name" to install the package
Expected Results:

RPM format package can be installed
Last Result **Not Run**
Keywords: None

Test Case TC-1429: rpm install dependency package
Summary: rpm command should report dependency when installing package
Steps: 1. Get a RPM package or build one on local machine, which should have run-time dependency. For example, mc RPM should depends on ncurses-terminfo 2. Run "rpm -ivh package_name" and check the output, for example "rpm -ivh mc.rpm*" should report the dependency on ncurses-terminfo
Expected Results: rpm command should report message when some RPM installation depends on other packages
Last Result **Not Run**
Keywords: None

Test Case TC-1430: rpm remove package
Summary: rpm command can remove package in system
Steps: 1. Launch terminal and run command "rpm -e package_name" to remove some package, for example, avahi
Expected Results: RPM package can be removed by command rpm
Last Result **Not Run**
Keywords: None

Test Case TC-1431: check rpm install/removal log file size
Summary: The case is to track log file size after rpm install/removal
Steps: 1. After system is up, check the log file size after rpm/zypper install/removal 2. for rpm, there will be some database files under /var/lib/rpm/, named as "__db.xxx" and there will be some log files under /var/lib/rpm/log, named as "log.xxxxxx". Each file will occupy about 10MB. 3. after several rpm/zypper install/removal, rpm will create several log files under /var/lib/rpm/log, which eat lots of system disk space.
Expected Results: there should be some method to keep rpm log in a small size

Last Result	**Not Run**
Keywords:	None

Test Case TC-1432: boot and install from USB	
Summary: boot and install image from usb stick	
Steps: 1. plugin usb which contains live image burned 2. configure device BIOS to firstly boot from USB if necessary 3. boot the device and select some option like "Boot and Install" from boot menu 4. proceed through default install process 5. Remove USB, and reboot into new installed system.	
Expected Results: 1. User can choose install system from usb stick onto harddisk from boot menu or command line option 2. Imstalled system can boot up	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1433: live boot from USB	
Summary: live boot from USB	
Steps: boot live image from usb stick 1. plugin usb which contains live image burned 2. configure device BIOS to firstly boot from USB if necessary 3. boot the device and select some option like "boot from usb" from boot menu	
Expected Results: 1. User can choose boot from live image on usb stick from boot menu or command line option 2. Live image can boot up with usb stick	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1434: boot from runlevel 3
Summary: Verify that system can boot from runlevel 3
Steps: 1. Boot into system and edit /etc/inittab to make sure system enter init 3 by default ######## id:3:initdefault

######## 2. reboot system, and press Tab to enter "grub" 3. edit "kernel" line and add "psplash=false text" at the end 4. Press "enter" to boot system
Expected Results: system should boot to runlevel 3.
Last Result **Not Run**
Keywords: None

Test Case TC-1435: boot from runlevel 5
Summary: Verify that system can boot from runlevel 5
Steps: 1. Boot into system and edit /etc/inittab to make sure system enter init 5 by default ######## id:5:initdefault ######## 2. reboot system, and press Tab to enter "grub" 3. edit "kernel" line and make sure no "psplash=false text" in grub cmdline 4. Press "enter" to boot system Note: The test is only for sato image.
Expected Results: system should boot to runlevel 5.
Last Result **Not Run**
Keywords: None

Test Case TC-1436: g++ compile in sdk image
Summary: check if g++ can compile program in sdk image
Steps: 1. Boot up sdk image 2. check if g++ is built in 3. compile following program test.c "g++ test.c -o test -lm" 4. run "test" and check the output test.c: ########## #include <stdio.h> #include <math.h> double

```
convert(long long l)
{
  return (double)l;    // or double(l)
}

int
main(int argc, char * argv[])
{
  long long l = 10;
  double f;

  f = convert(l);
  printf("convert: %lld => %f\n", l, f);

  f = 1234.67;
  printf("floorf(%f) = %f\n", f, floorf(f));
  return 0;
}
##########
```

Expected Results:

executable binary test can run without problem

Last Result	**Not Run**
Keywords:	None

Test Case TC-1437: gcc compile in sdk image

Summary:

check if gcc can compile program in sdk image

Steps:

1. Boot up sdk image
2. check if gcc is built in
3. compile following program test.c "gcc test.c -o test -lm"
4. run "test" and check the output

test.c:

```
##########
#include <stdio.h>
#include <math.h>

double
convert(long long l)
{
  return (double)l;    // or double(l)
}

int
main(int argc, char * argv[])
{
  long long l = 10;
  double f;

  f = convert(l);
  printf("convert: %lld => %f\n", l, f);

  f = 1234.67;
  printf("floorf(%f) = %f\n", f, floorf(f));
  return 0;
}
```

##########
Expected Results: executable binary test can run without problem
Last Result **Not Run**
Keywords: None

Test Case TC-1438: run command make in sdk image
Summary: check if command make can work in sdk image
Steps: 1. Boot up sdk image 2. check if make is built in 3. run command "make" with following makefile and build the test.c file from case "gcc compile in sdk image" test: test.o gcc -o test test.o -lm test.o: test.c gcc -c test.c
Expected Results: make command can work without problem
Last Result **Not Run**
Keywords: None

Test Case TC-1439: cvs project compile in sdk image
Summary: cvs project could be compiled in sdk image
Steps: 1. Download cvs project from http://ftp.gnu.org/non-gnu/cvs/source/feature/1.12.13/cvs-1.12.13.tar.bz2 2. Copy cvs tarball into sdk image 3. Extract the tarball and do "configure", "make" and "make install"
Expected Results: cvs project could be compiled successfully
Last Result **Not Run**
Keywords: None

Test Case TC-1440: iptables project compile in sdk image
Summary: iptables project could be compiled in sdk image
Steps:

1. Download iptables project from http://netfilter.org/projects/iptables/files/iptables-1.4.11.tar.bz2

2. Copy iptables tarball into sdk image
3. Extract the tarball and do "configure", "make" and "make install"

Expected Results:

iptables could be compiled successfully

Last Result	**Not Run**
Keywords:	None

Test Case TC-1441: sudoku-savant project compile in sdk image	
Summary:	
sudoku-savant could be compiled in sdk image	
Steps:	
1. Download sudoku-savant project from http://downloads.sourceforge.net/project/sudoku-savant/sudoku-savant/sudoku-savant-1.3/sudoku-savant-1.3.tar.bz2 2. Copy sudoku-savant tarball into sdk image 3. Extract the tarball and do "configure", "make"	
Expected Results:	
sudoku-savant could be compiled successfully	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1442: perl program work in image	
Summary:	
A perl program could be executed and output correctly in image	
Steps:	
1. Check if perl is installed in image and could run with "perl -v" 2. Prepare a perl program like followig test.pl 3. Run "perl test.pl" ######## $a = 9.01e+21 + 0.01 - 9.01e+21; print ("the value of a is ", $a, "\n"); $a = 9.01e+21 - 9.01e+21 + 0.01; print ("the value of a is ", $a, "\n"); ########	
Expected Results:	
The test.pl could run without problem	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1443: shutdown system	
Summary: verify that system can be shutdown by command	
Steps: 1. boot system 2. launch terminal and run "shutdown -h now" or "poweroff"	
Expected Results: System can be shutdown successfully	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1444: reboot system	
Summary: verify that system can boot by command	
Steps: 1. boot system 2. launch terminal and run "reboot"	
Expected Results: System can reboot successfully	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1445: adjust date and time	
Summary: adjust date and time	

Steps:
 	
1.launch terminal and run "date -R" to check current system time
2.adjust Date&Time by these commands:
For date command from coreutils, for example the sdk image use coreutils, you should use following syntax:
$ date -s "10:00:00 20100809"
$ date -R
$ Mon, 09 Aug 2010 10:00:00 +0000
For date command in busybox, for example the sato image use busybox, you should use following syntax:
$ date "080910002010"
$ date -R
$ Mon, 09 Aug 2010 10:00:00 +0000
3. check date with "date -R" and the time shown on matchbox-panel	

Expected Results: System time should be adjust to what you specified	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1446: switch among multi applications and desktop	
Summary: switch among multi applications and desktop	
Steps: 1. launch several applications(like contacts, file manager) 2. launch terminal 3. switch among multi applications and desktop 4. close applications Note: The case is for sato image only.	
Expected Results: 1. user could switch among multi applications and desktop	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1447: vncserver for target	
Summary: Check if vncserver setup work in target and vnc client could connect it	
Steps: 1. Check if x11vnc is installed in target 2. Run command "x11vnc -display :0.0", check the ip address of the target 3. On a client, run command "vncviewer $ip_address_of_target:0"	
Expected Results: A virtual X desktop of target should be pop-up on the client	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1448: file manager	
Summary: file manager	
Steps: 1.launch file manager from application panel 2.view folder/file in file manager 3.copy and paste folder/file in file manager Note: The test is only for sato image	
Expected Results: 1.folder and file could be listed in file browser with different display mode	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1449: system dmesg log check	
Summary: check if there is error in dmesg after system boot up	
Steps: 1. boot system and run command "dmesg"	
Expected Results: No error message in dmesg	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1450: usb mount	
Summary: verify that system can mount plugged usb automatically	
Steps: 1. boot system 2. plug usb stick	
Expected Results: 1. system notify that usb stick is accessible	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1451: usb read files	
Summary: verify that system can read files from usb	
Steps: 1. boot system 2. plug usb stick 3. view files in usb by file browser 4.copy some files from usb to local hardware	
Expected Results: 1. view/copy successfully	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1452: usb umount
Summary: verify that system can unmout usb automically

Steps: 1. boot system 2. plug usb stick 3. view files in usb by file browser 4.unplug usb	
Expected Results: 1. usb direcoty in file browser automatically missed	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1453: usb write files	
Summary: verify that system can write files to usb	
Steps: 1. boot system 2. plug usb stick 3. create files in usb 4.copy some files from local hardware to usb	
Expected Results: 1. create/copy successfully	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1454: file copy by scp	
Summary: check if file can be copied from remote machine to device by scp	
Steps: 1. check avahi is install and started 2. get system IP and try "scp file $IP:/home/root" from remote machine (file >= 500M for real HW, file>=5M for QEMU)	
Expected Results: File can be copied from remote machine to device by scp	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1455: connman launch after boot
Summary: After system booted, the connmand daemon should be launched
Steps: 1. boot system

2. "ps |grep connmand" 3. check if there is a thread named connmand in background
Expected Results: There should be one thread named connmand in background

Last Result	**Not Run**
Keywords:	None

Test Case TC-1456: ethernet enabled in connman
Summary: After system boot, ethernet can get IP address with connman
Steps: 1. boot system with network cable plugged in 2. "ps |grep connmand" if connmand is started 3. "ifconfig" check ethernet could get IP address and ping the address from remote machine
Expected Results: Ethernet interface can get IP via connman

Last Result	**Not Run**
Keywords:	None

Test Case TC-1457: only one connmand in background
Summary: there should be no more than one connmand in background
Steps: 1. boot system 2. "ps |grep connmand" 3. the connmand should be in background 4. run command "connmand" 5. check if the second connmand can be generated
Expected Results: There will be only one connmand instance in background

Last Result	**Not Run**
Keywords:	None

Test Case TC-1458: remote access by ssh
Summary: check if the device can be accessed remotely by ssh
Steps: 1. check avahi is install and started 2. get system IP and try "ssh $IP" from remote machine
Expected Results:

it is ok to access system by ssh from remote machine
Last Result **Not Run**
Keywords: None

Test Case TC-1459: ethernet static ip set in connman
Summary: we could set static ip for ethernet in connman
Steps: 1. launch connman-properities 2. choose ethernet device and set static ip for it. For example, in our internal network, we can set as following: ip address: 10.239.48.xxx Broadcast: 10.239.48.255 Mask: 255.255.255.0
Expected Results: we can set static ip for ethernet device
Last Result **Not Run**
Keywords: None

Test Case TC-1460: ethernet get IP in connman via DHCP
Summary: ethernet device can get IP in connman via DHCP
Steps: 1. Set static IP for ethernet device in connman 2. Check if ethernet device can work with static IP 3. Choose DHCP method for ethernet device 4. Check with ping if ethernet device get IP address via DHCP
Expected Results: Ethernet device can get dynamic IP address via DHCP in connman
Last Result **Not Run**
Keywords: None

Test Case TC-1461: connman offline mode in connman-gnome
Summary: change offline mode in comman-gnome can make all connection off
Steps:

1. Launch connman-properties after system booting

2. choose "offline mode" and check the connection of all network interfaces

Expected Results:

All connection should be off after clicking "offline mode"

Last Result	**Not Run**
Keywords:	None

Test Case TC-1462: X server can start up with runlevel 5 boot
Summary:
check if X server can work well after system runlevel 5 booting
Steps:
1. boot up system with default runlevel
Expected Results:
X server can start up well and desktop display has no problem

Last Result	**Not Run**
Keywords:	None

Test Case TC-1463: qt application quicky
Summary:
quicky is a simple note-taking application with Wiki-style syntax and behaviour
Steps:
launch quicky and write something in quicky
Expected Results:
http://qt-apps.org/content/show.php/Quicky?content=80325

Last Result	**Not Run**
Keywords:	None

Test Case TC-1464: standby
Summary:
system can enter standby and resume from standby
Steps:
1. boot system and launch terminal; check output of "date" and launch script "continue.sh" 2. echo "mem" > /sys/power/state 3. After system go into S3 mode, move mouse or press any key to make it resume 4. Check "date" and script "continue.sh" 5. Check if application in X can work as normal

continue.sh as below:

```
##################
#!/bin/sh

i=1
while [ 0 ]
do
 echo $i
 sleep 1
 i=$((i+1))
done
##################
```

Expected Results:

screen should resume back and script can run continuously

Last Result	**Not Run**
Keywords:	None

Test Case TC-1465: check CPU utilization after standby	
Summary:	
check CPU utilization after standby	
Steps:	
1. Start up system 2. run "top" command and check if there is any process eating CPU time 3. make system into standby and resume it 4. run "top" command and check if there is any difference with the data before standby	
Expected Results:	
There should be no big difference before/after standby with "top"	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1466: Test if LAN device works well after resume from suspend state	
Summary:	
Test if LAN device works well after resume from suspend state.	
Steps:	
1. boot system and launch terminal 2. echo "mem" > /sys/power/state 3. After system go into S3 mode, move mouse or press any key to make it resume 4. check ping status	
Expected Results:	
ping should always work before/after standby	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1467: Test if usb hid device works well after resume from suspend state	
Summary: Test if usb hid device works well after resume from suspend state.	
Steps: 1. boot system and launch terminal 2. echo "mem" > /sys/power/state 3. After system go into S3 mode, move mouse or press any key to make it resume 4. check usb mouse and keyboard	
Expected Results: usb mouse and keyboard should work	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1468: disk space check	
Summary: There should be enough disk space for QEMU rootfs	
Steps: 1. Launch QEMU targets(with rootfs.ext3 file) 2. Check the output of command df 3. If there is less than 5M disk space available, we assume it a failure	
Expected Results: There should be enough disk space for QEMU targets	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1469: click terminal icon on X desktop	
Summary: terminal icon should work without problem on X desktop	
Steps: 1. After system launch and X start up, click terminal icon on desktop 2. Check if only one terminal window launched and no other problem met	
Expected Results: there should be no problem after launching terminal	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1470: Add multiple files in music player
Summary: music player should be no problem when adding multiple files at same time

Steps: 1. Launch music player 2. Add multiple files(5 files) in music player at same time
Expected Results: music player should be OK with this action

Last Result	**Not Run**
Keywords:	None

Test Case TC-1471: system shutdown with UNFS
Summary: system shutdown with UNFS should work
Steps: 1. Use UNFS to start QEMU targets 2. Run shutdown in QEMU targets
Expected Results: QEMU shutdown with UNFS should work

Last Result	**Not Run**
Keywords:	None

Test Case TC-1472: no connman-gnome icon on desktop
Summary: there should be no connman-gnome icon on desktop
Steps: 1. Launch sato image 2. There should be no connman-gnome icon on desktop, and connman-properties should be only invoked by toolbar
Expected Results: There should be no connman-gnome icon on desktop, and connman-properties should be only invoked by toolbar

Last Result	**Not Run**
Keywords:	None

Test Case TC-1473: application contacts should work
Summary: application contacts should work without problem
Steps: 1. Make sure X is started up 2. Check if there is "contacts" icon on desktop and run it 3. Check if there is any error by checking the output of this action and dmesg log
Expected Results:

"contacts" launch should not cause any error
Last Result **Not Run**
Keywords: None

Test Case TC-1474: x11vnc icon click for target
Summary: Check if vncserver could work in target by clicking x11vnc icon
Steps: 1. Check if there is a x11vnc icon in target 2. Click the x11vnc icon and check the ip address of the target 3. On a client, run command "vncviewer $ip_address_of_target:0"
Expected Results: A virtual X desktop of target should be pop-up on the client
Last Result **Not Run**
Keywords: None

Test Case TC-1475: RTLDLIST path check for ldd command
Summary: check if the file set in RTLDLIST is valid
Steps: 1. After system is up, check if the RTLDLIST variable in ldd command 2. The file path set in RTLDLIST should be valid
Expected Results: check if the file set in RTLDLIST is valid
Last Result **Not Run**
Keywords: None

Test Case TC-1476: check bash in image
Summary: check if bash exists in image
Steps: 1. After system is up, check if bash command exists
Expected Results: bash command should exist in image
Last Result **Not Run**
Keywords: None

Test Case TC-1477: "Install/Remove Software" icon should work	
Summary: "Install/Remove Software" icon should work	
Steps: 1. After system is up, check if "Install/Remove Software" icon could work	
Expected Results: "Install/Remove Software" icon should work	
Last Result	**Not Run**
Keywords:	None

1.2 Test Suite : ADT

Test Case TC-1478: gcc from ADT toolchain can build c program
Summary: gcc from ADT toolchain can build c program and run with qemu-${ARCH} command or in target image

Steps:
 	
1. Install toolchain tarball and setup cross compile environment
2. compile following program test.c "${CC} test.c -o test -cc -lm"
3. run "test" with qemu-${ARCH} or run it into corresponding target image and check the output
 	
Note: Currently, only i586_i586, x86-64_x86-64 and i586_$X(x is mips, arm and ppc) toolchain tarballs are covered in testing.
 	
#########
#include <stdio.h>
#include <math.h>
 	
double
convert(long long l)
{
  return (double)l; // or double(l)
}
 	
int
main(int argc, char * argv[])
{
  long long l = 10;
  double f;
 	
  f = convert(l);
  printf("convert: %lld => %f\n", l, f);
 	
  f = 1234.67;
  printf("floorf(%f) = %f\n", f, floorf(f));
  return 0;
}
#########

Expected Results:
executable binary test can run without problem

Last Result	**Not Run**
Keywords:	None

Test Case TC-1479: g++ from ADT toolchain can build c program
Summary:
g++ from ADT toolchain can build c program and run with qemu-${ARCH} command or in target image
Steps:

1. Install toolchain tarball and setup cross compile environment
2. compile following program test.c "${CXX} test.c -o test -cc++ -lm"
3. run "test" with qemu-${ARCH} or run it in corresponding target image and check the output
 	
Note: Currently, only i586_i586, x86-64_x86-64 and i586_$X(x is mips, arm and ppc) toolchain tarballs are covered in testing.
 	
#########
#include <stdio.h>
#include <math.h>
 	
double
convert(long long l)
{
  return (double)l; // or double(l)
}
 	
int
main(int argc, char * argv[])
{
  long long l = 10;
  double f;
 	
  f = convert(l);
  printf("convert: %lld => %f\n", l, f);
 	
  f = 1234.67;
  printf("floorf(%f) = %f\n", f, floorf(f));
  return 0;
}
#########

Expected Results:
executable binary test can run without problem

Last Result	**Not Run**
Keywords:	None

Test Case TC-1480: ADT toolchain could build cvs project
Summary:
ADT toolchain could build cvs project

Steps: 1. Install toolchain tarball and setup cross compile environment 2. Download cvs project, http://ftp.gnu.org/non-gnu/cvs/source/feature/1.12.13/cvs-1.12.13.tar.bz2 3. With the cross compile environment, run "./configure ${CONFIGURE_FLAGS}", "make", "make install DESTDIR=/opt/tmp" Note: Currently, only i586_i586, x86-64_x86-64 and i586_$X(x is mips, arm and ppc) toolchain tarballs are covered in testing.
Expected Results: cvs project could be compiled successfully with ADT toolchain

Last Result	**Not Run**
Keywords:	None

Test Case TC-1481: ADT toolchain could build iptables project
Summary: iptables project could be compiled with ADT toolchain
Steps: 1. Install toolchain tarball and setup cross compile environment 2. Download iptables project, http://netfilter.org/projects/iptables/files/iptables-1.4.11.tar.bz2 3. With the cross compile environment, run "./configure ${CONFIGURE_FLAGS}", "make", "make install DESTDIR=/opt/tmp" Note: Currently, only i586_i586, x86-64_x86-64 and i586_$X(x is mips, arm and ppc) toolchain tarballs are covered in testing.
Expected Results: iptables could be compiled successfully

Last Result	**Not Run**
Keywords:	None

Test Case TC-1482: ADT toolchain could build sudoku-savant project
Summary: sudoku-savant could be compiled with ADT toolchain
Steps: 1. Install toolchain tarball and setup cross compile environment 2. Download sudoku-savant project, http://downloads.sourceforge.net/project/sudoku-savant/sudoku-savant/sudoku-savant-1.3/sudoku-savant-1.3.tar.bz2 3. With the cross compile environment, run "./configure ${CONFIGURE_FLAGS}", "make", "make install DESTDIR=/opt/tmp" Note: Currently, only i586_i586, x86-64_x86-64 and i586_$X(x is mips, arm and ppc) toolchain tarballs are covered in testing.
Expected Results: sudoku-savant could be compiled successfully

Last Result	**Not Run**

Keywords:	None

Test Case TC-1483: unfs support for qemu target	
Summary: Check if unfs works for qemu target	
Steps: 1. Prepare a *rootfs.tar.bz2 image 2. Prepare a folder under poky directory as <rootfs-dir>, for example poky/temp 3. Run command "runqemu-extract-sdk *rootfs.tar.bz2 poky/temp" 4. Run command "runqemu nfs <kernel> <rootfs-dir>"	
Expected Results: QEMU target should be started with unfs	
Last Result	**Not Run**
Keywords:	None

1.3 Test Suite : Stress

Test Case TC-1484: crashme for stress	
Summary: Run crashme in real hardware for stress testing	
Steps: 1. Get crashme from http://people.delphiforums.com/gjc/crashme.html 2. By following the setup steps on above URL, build crashme in target. 3. Run crashme for 24 hours	
Expected Results: target should not crash with the program	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1485: helltest for stress
Summary: Run helltest for stress in target
Steps: 1. helltest is stress test suite, which does compiler test for hours 2. We download the test suite and run it for 24 hours
Expected Results: helltest should not make target crash

Last Result	**Not Run**
Keywords:	None

Test Case TC-1486: ltp for stress
Summary: Run ltp stress in real hardware for stress testing
Steps: LTP download: http://sourceforge.net/projects/ltp/files/LTP%20Source/ltp-20101031/ltp-full-20101031.bz2/download build steps: refer to http://ltp.sourceforge.net Run steps: 1. Build LTP with toolchain or in sdk image 2. Copy LTP folder into target, for example, /opt/ltp. Modify script "testscripts/ltpstress.sh", set “lostat=1”, ”NO_NETWORK=1” 3. cd testscripts/ && ./ltpstress.sh 4. This stress case will run for 24 hours
Expected Results: Check the result, target should not crash with the program.

Last Result	**Not Run**
Keywords:	None

1.4 Test Suite : Power/Performance

Test Case TC-1487: boot time collection
Summary: To collect boot time of clean installation, from grub to full desktop
Steps: 1. Reboot testing device at least 3 times and do not plug anything while collecting boot time by stopwatcher: #reboot
Expected Results: Provide average boot time and dmesg log

Last Result	**Not Run**
Keywords:	None

Test Case TC-1488: memory footprint
Summary:

collect data of the used/free memory
Steps: With default installtion, launch terminal and type 'free' to read the used/free disk space
Expected Results: Provide 'free' output

Last Result	**Not Run**
Keywords:	None

Test Case TC-1489: powertop log
Summary: collect powertop data
Steps: 1. Run "powertop -d" and record output 2. Save the percentage of deepest C state(C3 or C2)
Expected Results: Provide powertop output

Last Result	**Not Run**
Keywords:	None

Test Case TC-1490: Idle power consumption
Summary: Collect idle power consumption of target system
Steps: 1. Use power meter to collect ilde power consumption of target system for 10 minutes 2. Save it and compare it with old data
Expected Results: There should be no regression between old and new ilde power data

Last Result	**Not Run**
Keywords:	None

Test Case TC-1491: core build time for sato image
Summary: collect the core build time for sato qemux86 image
Steps: 1. Perpare a system with following configuration CPU: 4-core * 2-threads Intel(R) Core(TM) i7 CPU 860 @ 2.80GHz

Memory: 4GB Harddisk: 1TB OS: Ubuntu 10.04 x86_64 Kernel: 2.6.32-21 2. Download poky tree and make sure all the source packages have been downloaded 3. Build a qemux86 sato image and collect the time
Expected Results: There should be no regression for build time

Last Result	**Not Run**
Keywords:	None

1.5 Test Suite : Graphics

Test Case TC-1492: Graphics ABAT

Summary:

Yocto on SugarBay should pass Intel graphics ABAT testing

Steps:

1. Download ABAT test suite from internal git repository, git clone git://tinderbox.sh.intel.com/git/abat
2. Apply following patch to make it work on yocto environment
3. Run "./abat.sh" to run ABAT test

########

```
diff --git a/glxgears_check.sh b/glxgears_check.sh
index 17622b8..c4d3b97 100755
--- a/glxgears_check.sh
+++ b/glxgears_check.sh
@@ -31,7 +31,7 @@ else

     sleep 6

-    XPID=$( ps ax | awk '{print $1, $5}' | grep glxgears | awk '{print $1}')
+    XPID=$( ps | awk '{print $1, $5}' | grep glxgears | awk '{print $1}')
     if [ ! -z "$XPID" ]; then
         kill -9 $XPID >/dev/null 2>&1
         echo "glxgears can run, PASS!"
diff --git a/x_close.sh b/x_close.sh
index e287be1..3429f1a 100755
--- a/x_close.sh
+++ b/x_close.sh
@@ -22,7 +22,7 @@
 #
 function close_proc(){
 echo "kill process Xorg"
-XPID=$(  ps ax | awk '{print $1, $5}' | egrep "X$|Xorg$" | awk '{print $1}')
+XPID=$(  ps | awk '{print $1, $6}' | egrep "X$|Xorg$" | awk '{print $1}')
 if [ ! -z "$XPID" ]; then
     kill $XPID
     sleep 4
diff --git a/x_start.sh b/x_start.sh
index 9cf6eab..2305796 100755
--- a/x_start.sh
```

```
+++ b/x_start.sh
@@ -24,7 +24,7 @@
 X_ERROR=0

 #test whether X has started
-PXID=$(ps ax |awk '{print $1,$5}' |egrep "Xorg$|X$" |grep -v grep | awk '{print $1}')
+PXID=$(ps |awk '{print $1,$6}' |egrep "Xorg$|X$" |grep -v grep | awk '{print $1}')
 if [ ! -z "$PXID" ]; then
     echo "[WARNING] Xorg has started!"
     XORG_STATUS="started"
@@ -35,9 +35,11 @@ else
     #start up the x server
     echo "Start up the X server for test in display $DISPLAY....."

-    $XORG_DIR/bin/X >/dev/null 2>&1 &
+    #$XORG_DIR/bin/X >/dev/null 2>&1 &
+    #sleep 8
+    #xterm &
+    /etc/init.d/xserver-nodm start &
     sleep 8
-    xterm &
 fi
     XLOG_FILE=/var/log/Xorg.0.log
     [ -f $XORG_DIR/var/log/Xorg.0.log ] && XLOG_FILE=$XORG_DIR/var/log/Xorg.0.log
@@ -54,7 +56,7 @@ fi
         X_ERROR=1
     fi

-    XPID=$( ps ax | awk '{print $1, $5}' | egrep "X$|Xorg$" |grep -v grep| awk '{print $1}')
+    XPID=$( ps | awk '{print $1, $6}' | egrep "X$|Xorg$" |grep -v grep| awk '{print $1}')
     if [ -z "$XPID" ]; then
         echo "Start up X server FAIL!"
  echo
########
```

Expected Results:

All ABAT test should pass

Last Result	**Not Run**
Keywords:	None

Test Case TC-1493: openarena - 3D

Summary:

Run opernarena testing and compare the result with upstream graphics result

Steps:

1. Download and build openarena through phoronix test suite. first download a new phoronix from its website, then download the game in it. The openarena we use is v0.8.5.

```
####
phoronix-test-suite list-tests
phoronix-test-suite install openarena
####
```

2. Run the test suite with following command

```
####
vblank_mode=0 openarena +exec pts +set r_mode -1 +set r_fullscreen 1 +set r_customWidth $VIDEO_WIDTH +set r_customHeight $VIDEO_HEIGHT
####
```

The VIDEO_WIDTH and VIDEO_HEIGHT set the game's resolution，you can get current resolution by command "xrandr"

Expected Results:

Compare the result of Yocto with upstream graphics
Last Result **Not Run**
Keywords: None

Test Case TC-1494: urbanterror - 3D
Summary: Run urbanterror and compare the result of Yocto with upstream graphics
Steps: 1. download and build: This game also can get through phoronix-test-suite. 2. we should set some environments as following before test: ### OS_TYPE=Linux OS_ARCH=`uname -i` LOG_FILE=${LOGNOW_DIR}/${LOG_FILE} ### 3. Run urbanterror with following command ### vblank_mode=0 ./urbanterror +timedemo 1 +set demodone 'quit' +set demoloop1 'demo pts1; set nextdemo vstr demodone' +vstr demoloop1 +set r_customwidth $VIDEO_WIDTH +set r_customheight $VIDEO_HEIGHT ###
Expected Results: Get the FPS data of Yocto and compare it with upstream graphics
Last Result **Not Run**
Keywords: None

Test Case TC-1495: x11perf - 2D
Summary: Get fps data of x11per running
Steps: 1. Run "x11perf -aa10text" and "x11perf -rgb10text" 2. Get the FPS result and compare it with upstream graphics data on Sandybridge
Expected Results: There should not be big regression between Yocto and upstream linux
Last Result **Not Run**
Keywords: None

1.6 Test Suite : Mulitimedia

Test Case TC-1496: sound on/off	
Summary: check if sound can be turned on/off	
Steps: 1. copy amixer is installed 2. Run "amixer set Master on" to turn on audio device 3. Run "amixer set Master 64" to adjust to maxium volumn 4. Run "amixer set Speaker on" to turn on speaker 5. Run "amixer set Speaker 64" to adjust to maxium volumn 6. Run "amixer set Master off" to turn off audio device 7. Run "amixer set Speaker off" to turn off speaker	
Expected Results: Above commands can run without problem	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1497: audio play (mp3)	
Summary: make sure music player cannot play mp3 format file	
Steps: 1. copy sample mp3 file to system 2. launch music player and make sure it cannot play the mp3 file	
Expected Results: mp3 file can not be played	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1498: audio play (ogg)	
Summary: check if music player can play ogg format file	
Steps: 1. copy sample ogg file to system 2. launch music player can play the ogg file	
Expected Results: ogg file can be played without problem	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1499: audio stop (ogg)
Summary:

check if music player can play ogg format file
Steps: 1. copy sample ogg file to system 2. launch music player can play the ogg file 3. click "stop" button to stop playing 4. click "start" button to resume playing
Expected Results: ogg file can be start/stop without problem

Last Result	**Not Run**
Keywords:	None

Test Case TC-1500: audio play (wav)
Summary: check if music player can play wav format file
Steps: 1. copy sample wav file to system 2. launch music player can play the wav file
Expected Results: wav file can be played without problem

Last Result	**Not Run**
Keywords:	None

Test Case TC-1501: audio stop (wav)
Summary: check if music player can stop playing with wav format file
Steps: 1. copy sample wav file to system 2. launch music player can play the wav file 3. click "stop" button to stop playing 4. click "start" button to resume playing
Expected Results: wav file can be start/stop without problem

Last Result	**Not Run**
Keywords:	None

Test Case TC-1502: video play (mpeg)
Summary: make sure video player cannot play mpeg format file
Steps:

1. copy sample mpeg file to system 2. launch video player and make sure it cannot play the mpeg file
Expected Results: mpeg file cannot be played
Last Result **Not Run**
Keywords: None

Test Case TC-1503: video play (ogg)
Summary: check if video player can play ogg format file
Steps: 1. copy sample ogg file to system 2. launch video player can play the ogg file
Expected Results: ogg file can be played without problem
Last Result **Not Run**
Keywords: None

Test Case TC-1504: video stop (ogg)
Summary: check if video player can play ogg format file
Steps: 1. copy sample ogg file to system 2. launch video player can play the ogg file 3. click "stop" button to stop playing 4. click "start" button to resume playing
Expected Results: ogg file can be start/stop without problem
Last Result **Not Run**
Keywords: None

1.7 Test Suite : Compliance

Test Case TC-1505: LTP subset test suite
Summary: LTP subset test suite

Steps: For real hardware, run following component, syscalls fs fsx dio io mm ipc sched math nptl pty admin_tools timers commands For QEMU, run following component syscalls mm ipc sched math nptl pty admin_tools commands Run Instructions: LTP download: http://sourceforge.net/projects/ltp/files/LTP%20Source/ltp-20110915/ltp-full-20110915.bz2/download build steps: refer to http://ltp.sourceforge.net Run steps: 1. Build LTP with toolchain or in sdk image 2. For QEMU, create the qemu target with "-m 512", which makes some memory stress cases pass. For some issues, we could only set 128M for qemuarm and 256M for qemumips. 3. Copy LTP folder into target, for example, /opt/ltp. Modify script "runltp", remove test suites not to be tested 4. Comment runtests/sched: hackbench, which is not suitable to run in emulators 5. Comment creat08, oom01, oom02, oom03, oom04, which consume lots of memory 6. Prepare a tmp folder under your ltp folder, for example, create a tmp folder under your ltp folder, like /opt/ltp/tmp 7. ./runltp -p -l result-M2-20101218.log -C result-M2-20101218.fail -d /opt/ltp/tmp &> result-M2-20101218.fulllog (assume you mount your LTP disk at /opt and create your own tmp dir at /opt/ltp/tmp)
Expected Results: Check the result on wiki, https://wiki.yoctoproject.org/wiki/LTP_result, there should be no regression failure met.

Last Result	**Not Run**
Keywords:	None

Test Case TC-1506: POSIX subset test suite
Summary: Run subset test suite of POSIX test suite
Steps:

Get POSIX test suite from latest source Run steps: 1. cd open_posix_testsuite/ 2. make generate-makefiles 3. make conformance-all 4. make conformance-test 5. make tools-all 6. ./bin/run-posix-option-group-test.sh [AIO | MEM |MSG | SEM | SIG | THR | TMR | TPS]
Expected Results: Compare the test result on wiki, https://wiki.yoctoproject.org/wiki/Posix_result, there should be no more regression failures met.

Last Result	**Not Run**
Keywords:	None

Test Case TC-1507: LSB subset test suite
Summary: Run LSB subset test suite in target
Steps: 1. Get LSB image and start the image(if it is QEMU) with option "-m 512M" 2. Get the LSB test suite or run script creat-lsb-image under poky source directory "scripts/creat-lsb-image" 3. Setup environment for lsb image in target with script LSB_Setup.sh, it could be found under poky source directory "/meta/recipes-extended/lsb/lsbsetup/LSB_Setup.sh" 4. Select LSB test items in LSB web interface and run them
Expected Results: Check the result on wiki, https://wiki.pokylinux.org/wiki/index.php?title=LSB_result&action=edit&redlink=1. No regression failures should be met.

Last Result	**Not Run**
Keywords:	None

1.8 Test Suite : Core Build System

Test Case TC-1387: libva check (ogg video play)
Summary: check if libva is installed and used when video player playing ogg video file
Steps: 1. check if libva is installed on system 2. copy sample ogg file to system 3. launch video player can play the ogg file
Expected Results:

ogg file can be played without problem when libva is used

Last Result	**Not Run**
Keywords:	None

Test Case TC-1508: Init scripts
Summary: Provide an image/recipe skeleton as a canonical example. Check if can be built and run correctly
Steps: 1. Build image from poky source, check if skeleton script and skeleton-test can be built into the image a. download poky source b. modify the line IMAGE_FEATURES += "apps-console-core ${SATO_IMAGE_FEATURES}" to IMAGE_FEATURES += "apps-console-core ${SATO_IMAGE_FEATURES}} service" in meta/recipes-sato/images/core-image-sato.bb (for sato image) or core-image-sato-sdk.bb (for sato-sdk image) c. $ source oe-init-build-env add line “<POKY_BASE>/meta-skeleton \” to conf/bblayer.conf d. build the image e. boot up the image, check the skeleton and skeleton-test should be in right place /etc/init.d/skeleton /usr/sbin/skeleton-test 2. Verify the basic function of skeleton. Check if skeleton script can start/stop the skeleton-test daemon.
Expected Results: Init scripts can be built and run correctly
Last Result **Not Run**
Keywords: None

Test Case TC-1386: lib32 connman-gnome built for qemux86-64 - ipk
Summary: build lib32 connman-gnome and include it in qemux86-64 image
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set

MACHINE to qemux86-64 3. set "IMAGE_INSTALL_append = "lib32-connman-gnome"" 4. with ipk set for package format, build core-sato image 5. after build finished, start up the image and check if connman and related packages are 32-bit
Expected Results: user could build lib32 connman-gnome and include it in qemux86-64 image

Last Result	**Not Run**
Keywords:	None

Test Case TC-1509: Minimal image
Summary: Check if the minimal image can be built and run correctly.
Steps: 1. Build a minimal image from poky source by following the wiki: https://wiki.yoctoproject.org/wiki/Minimal_Image 2. Check the size of the image. It should take less than 5M disk space after extraction. 3. Verify the basic function of the image. Run “busybox –list” to get the commands list. Check if these commands can run correctly.
Expected Results: The minimal image can be built and run correctly.

Last Result	**Not Run**
Keywords:	None

Test Case TC-1385: lib32 connman-gnome built for qemux86-64 - rpm
Summary: build lib32 connman-gnome and include it in qemux86-64 image
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86-64 3. set "IMAGE_INSTALL_append = "lib32-connman-gnome"" 4. with rpm set for package format, build core-sato image 5. after build finished, start up the image and check if connman and related packages are 32-bit
Expected Results: user could build lib32 connman-gnome and include it in qemux86-64 image

Last Result	**Not Run**
Keywords:	None

Test Case TC-1510: Share gcc work directories
Summary: This feature make gcc use the shared source directory during the different building. Check if this

feature can work for gcc 4.5.1 and gcc 4.6.0.
Steps: 1. Download the poky source and set build environment. 2. For gcc 4.5.1, add 2 lines to conf/local.conf : GCCVERSION ?= "4.5.1" SDKGCCVERSION ?= "4.5.1" For gcc 4.6.1, there is no need to add these 2 lines to conf/local.conf 3. Run bitbake command as below: bitbake gcc-cross bitbake gcc-cross gcc-cross-initial gcc-cross-intermediate -c clean bitbake gcc-crosssdk bitbake gcc-runtime bitbake libgcc bitbake gcc-cross-canadian-arm (for arm arch) bitbake gcc-cross-canadian-powerpc (for ppc arch) bitbake gcc-cross-canadian-mips (for mips arch) 4. Run "bitbake core-image-minimal", "bitbake core-image-sato", "bitbake core-image-sato-sdk" to build images. Verify the basic function of the images.
Expected Results: After step3, you can check the tmp/work-shared/gcc-4.6.0 or tmp/work-shared/gcc-4.5.1 should in the build directory. Check the time of build process and the disk space usage of tmp/work-shared/gcc-version sub-directory. The images should be built and can work correctly.

Last Result	**Not Run**
Keywords:	None

Test Case TC-1511: ccache as native tool
Summary: ccache - a fast C/C++ compiler cache.
Steps: 1. Make sure the native ccache is not installed on local machine and compile 'less' bbfile without native ccache support. bitbake ccache-native -c clean bitbake less -c clean bitbake less -c compile Check the compile log under .../tmp/work/mips-poky-linux/less-443-r0/temp/log.do_compile 2. Build native tool 'ccache' bitbake ccache-native Check the ccache-native installed location ..tmp/sysroots/x86_64-linux/usr/bin/ccache 3. Compile less bbfile again with native ccache support bitbake less -c clean bitbake less -c compile Check the compile with ccache log under .../tmp/work/mips-poky-linux/less-443-r0/temp/log.do_compile. The native ccache should be used when compiled.
Expected Results: The ccache-native should be built successfully and be installed to the correct location. The ccache-navive will be used when compile file.

Last Result	**Not Run**
Keywords:	None

Test Case TC-1512: PAM support	
Summary: Check the Yocto should support PAM (Pluggable Authentication Module)	
Steps: 1. Build a sato-sdk image from poky source with PAM support by following the wiki: https://wiki.yoctoproject.org/wiki/PAM_Integration 2. Refer to https://wiki.yoctoproject.org/wiki/PAM_Integration , check the commands 'dropbear', 'login', 'passwd', 'useradd', 'su' can work correctly with PAM support and verify the function of PAM.	
Expected Results: The commands which have PAM support should run correctly and the function of PAM should work without problems.	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1513: kernel interactive targets	
Summary: Check if yocto can support kernel interactive target build	
Steps: 1. download yocto source tree 2. prepare yocto build environment 3. Run "bitbake linux-yocto -c menuconfig" 4. Check if a new bash terminal pop up and menuconfig can be triggered	
Expected Results: menuconfig for kernel can be triggered with yocto build command	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1514: KVM enabled with qemu	
Summary: qemu can be started with KVM enabled	
Steps: 1. build a kernel with KVM enabled 2. Start qemu with option "kvm" with runqemu 3. Check if qemu starts up and if kvm_intel is used 4. If kvm_intel is not used when starting qemu, it will shows 0 in "Used by" column when you run "lsmod | grep kvm_intel"	
Expected Results: KVM enabled with qemu	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1515: non-GPLv3 build check	
Summary: Check if non-GPLv3 build could pass and it does not has any GPLv3 packages installed	

Steps:
 	
1. Set following sentences in local.conf to GPLv3
#####
INCOMPATIBLE_LICENSE = "GPLv3"
#####
2. Build core-image-minimal and core-image-basic
3. Start up target after build is finished
4. Run following script to check if any GPLv3 packages installed, some packages are GPLv3 exception, like libgcc1, libstdc++ and less.
 	
##################
#!/bin/sh
 	
temp=`mktemp`
rpm -qa > $temp
ret=0
 	
for i in `cat $temp`
do
  rpm -qi $i | grep License | grep -i gplv3 > /dev/null 2>&1
  if [$? -eq 0]; then
  license=`rpm -qi $i | grep License | awk -F"License:" '{print $2}'`
  echo "package $i has inconsistent license: $license"
  ret=1
  fi
done
 	
rm -rf $temp
exit $ret
####################	

Expected Results: non-GPLv3 build pass and no GPLv3 packages installed in the image	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1516: yocto build in Fedora 15	
Summary: Build latest yocto in x86_64 Fedora 15 host	
Steps: 1. By following the yocto handbook, download latest yocto source 2. Build core-image-minimal on Fedora 15	
Expected Results: Yocto build should pass on Fedora 15	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1517: yocto build in OpenSuse 11.4	
Summary: Build latest yocto in x86_64 OpenSuse 11.4	
Steps: 1. By following the yocto handbook, download latest yocto source 2. Build core-image-minimal on OpenSuse 11.4	
Expected Results: Build should pass on OpenSuse 11.3	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1518: yocto build in Ubuntu 11.04	
Summary: Build latest yocto in x86_64 Ubuntu 11.04	
Steps: 1. By following the yocto handbook, download latest yocto source 2. Build core-image-minimal on Utuntu 11.04	
Expected Results: Yocto build should pass on Utuntu 10.04	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1519: yocto build in KVM	
Summary: Build yocto in KVM should work	
Steps: 1. Setup a VM environment with KVM enabled, for example, RHEL6 2. Prepare a VM for yocto build testing, for example, OpenSuse 11.3 3. By following the yocto handbook, download latest yocto source into the VM 4. Build core-image-minimal in the VM	
Expected Results: Yocto build in VM should work same as in real host	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1520: sstate work on local host
Summary: Check if sstate could work with local cache
Steps:

1. Follow the wiki steps to setup a sstate cache on local machine, https://wiki.yoctoproject.org/wiki/Enable_sstate_cache
2. Prepare another yocto source directory and set the SSTATE_DIR the cache you setup in step 1)
3. Run poky build, for example, "bitbake core-image-minimal". You should note following things if sstate works:

########
NOTE: Preparing runqueue
NOTE: Executing SetScene Tasks
NOTE: Running setscene task 118 of 155 (virtual:native:/home/lulianhao/poky-build/edwin/poky/meta/recipes-devtools/pseudo/pseudo_git.bb:do_populate_sysroot_setscene)
NOTE: Running setscene task 119 of 155 (/home/lulianhao/poky-build/edwin/poky/meta/recipes-devtools/quilt/quilt-native_0.48.bb:do_populate_sysroot_setscene
########

Expected Results:

sstate should work and reduce build time

Last Result	**Not Run**
Keywords:	None

Test Case TC-1521: gcc set to 4.5.1 for core build
Summary:
gcc related options should be set to 4.5.1 for 4.5.1 build
Steps:
1. Download poky source and prepare the build environment 2. Set GCCVERSION and SDKGCCVERSION to 4.5.1 in meta/conf/distro/include/tcmode-default.inc 3. Run "bitbake -s | grep gcc" and check the output, all gcc related options should be set to 4.5.1
Expected Results:
all gcc related options should be set to 4.5.1

Last Result	**Not Run**
Keywords:	None

Test Case TC-1522: btrfs format image build
Summary:
btrfs format image could be built out
Steps:
1. set IMAGE_FSTYPES = "btrfs" and KERNEL_FEATURES_append = " cfg/btrfs " in local.conf 2. build a core-image-minimal image, the image should be btrfs format
Expected Results:
btrfs format image could be built out

Last Result	**Not Run**
Keywords:	None

Test Case TC-1523: btrfs format image boot up	
Summary: btrfs format image could be booted up	
Steps: 1. set IMAGE_FSTYPES = "btrfs" and KERNEL_FEATURES_append = " cfg/btrfs " in local.conf 2. build a qemux86 core-image-minimal image and boot up it	
Expected Results: btrfs format image could be booted up	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1524: lib64-zlib lib32-zlib build	
Summary: lib64-zlib lib32-zlib build should pass with multilib enabled	
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build 3. build lib64-zlib and lib32-zlib, which should build pass without error	
Expected Results: lib64-zlib lib32-zlib build should pass with multilib enabled	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1525: lib32 sato image build - qemux86	
Summary: lib32 sato image could be built out with multilib support	
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86 3. with rpm set for package format, build lib32 core-sato image 4. after build finished, start up the image and check if all app are 32-bit, kernel with 32-bit	
Expected Results: lib32 sato image could be built out with multilib support	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1526: lib32 sato image build - qemux86-64
Summary: lib32 sato image could be built out with multilib support

Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86 3. with rpm set for package format, build lib32 core-sato image 4. after build finished, start up the image and check if all app are 32-bit, kernel with 64-bit
Expected Results: lib32 sato image could be built out with multilib support

Last Result	**Not Run**
Keywords:	None

Test Case TC-1527: lib64 sato image build - qemux86
Summary: lib64 sato image should be built out with multilib support
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86 3. with rpm set for package format, build lib64 core-sato image 4. after build finished, start up the image and check if all app are 64-bit, kernel with 32-bit
Expected Results: lib64 sato-sdk image should be built out with multilib support

Last Result	**Not Run**
Keywords:	None

Test Case TC-1528: lib64 sato image build - qemux86-64
Summary: lib64 sato image should be built out with multilib support
Steps: 1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86 3. with rpm set for package format, build lib64 core-sato image 4. after build finished, start up the image and check if all app are 64-bit, kernel with 64-bit
Expected Results: lib64 sato-sdk image should be built out with multilib support

Last Result	**Not Run**
Keywords:	None

Test Case TC-1529: lib64 sato image build - qemux86-64/ipk
Summary:

lib64 sato image should be built out with multilib support

Steps:

1. Prepare poky build environment
2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86
3. with ipk set for package format, build lib64 core-sato image
4. after build finished, start up the image and check if all app are 64-bit, kernel with 64-bit

Expected Results:

lib64 sato-sdk image should be built out with multilib support

Last Result	**Not Run**
Keywords:	None

Test Case TC-1530: lib64 sato image build - qemux86-64/deb
Summary:
lib64 sato image should be built out with multilib support
Steps:
1. Prepare poky build environment 2. by following https://wiki.pokylinux.org/wiki/Multilib, set local.conf to enable multilib build and set MACHINE to qemux86 3. with deb set for package format, build lib64 core-sato image 4. after build finished, start up the image and check if all app are 64-bit, kernel with 64-bit
Expected Results:
lib64 sato-sdk image should be built out with multilib support

Last Result	**Not Run**
Keywords:	None

Test Case TC-1534: bitbake-layers show_layers
Summary:
show_layers could show current layers
Steps:
1. prepare poky build environment 2. add meta-rt into bblayer.conf 3. run "bitbake-layers show_layers", it should show the layers defined in bblayer.conf
Expected Results:
show_layers could show current layers

Last Result	**Not Run**
Keywords:	None

Test Case TC-1535: bitbake-layers show_overlayed
Summary:

overlayed recipes should be shown with bitbake-layers
Steps: 1. prepare poky build environment 2. copy a recipe from meta layer into meta-yocto, for example, /home/jxu49/osel/poky/meta/recipes-graphics/clutter/clutter-1.6_1.6.14.bb 3. run "bitbake-layers show_overlayed", it should report clutter is overlayed by meta-yocto
Expected Results: overlayed recipes should be shown with bitbake-layers

Last Result	**Not Run**
Keywords:	None

Test Case TC-1536: bitbake-layers show_appends
Summary: bitbake-layers show_appends should list bbappend files and recipe files they apply to
Steps: 1. prepare poky build environment 2. run "bitbake-layers show_appends", it should list bbappend files and recipe files they apply to
Expected Results: bitbake-layers show_appends should list bbappend files and recipe files they apply to

Last Result	**Not Run**
Keywords:	None

Test Case TC-1537: bitbake-layers flatten
Summary: bitbake-layers flattens layer configuration into a separate output directory
Steps: 1. prepare poky build environment 2. create a folder, for example, test 3. run "bitbake-layers flatten test", all contents of all layers should be moved into the test folder, with any bbappends appended to corresponding recipes 4. check if bbappends take effect, for example, check if test/recipes-bsp/formfactor/formfactor_0.0.bb has the code defined in meta-yocto/recipes-bsp/formfactor/formfactor_0.0.bbappend
Expected Results: bitbake-layers flattens layer configuration into a separate output directory

Last Result	**Not Run**
Keywords:	None

Test Case TC-1388: buildhistory enable for yocto build
Summary:

check if buildhistory could work during yocto build
Steps: 1. build of an image (e.g. core-image-minimal) runs through successfully with it enabled (i.e. with INHERIT += "buildhistory" in local.conf). 2. Once a build with package history enabled has finished, verify that the output can be found in TMPDIR/pkghistory.
Expected Results: package information should be under TMPDIR/buildhistory

Last Result	**Not Run**
Keywords:	None

Test Case TC-1389: buildhistory error if do_package backwards
Summary: check if buildhistory reports error if PR of some recipes go backwards
Steps: 1. build some recipes and get the buildhistory log(for example, recipe "man") 2. change the PR of man backwards, for example from "r1" to "r0" 3. re-build the recipe
Expected Results: pkghistory reports error if PR of some recipes go backwards

Last Result	**Not Run**
Keywords:	None

Test Case TC-1393: minimal build with self-hosted-image on sugarbay
Summary: check if self-hosted-image could pass minimal build on sugarbay
Steps: 1. Get self-hosted-image source code and build out the image for sugarbay 2. install the image on sugarbay 3. get yocto source on sugarbay and build a minimal image on it
Expected Results: self-hosted-image should pass minimal build on sugarbay

Last Result	**Not Run**
Keywords:	None

Test Case TC-1538: x32 image build
Summary: x32 image could be built out successfully
Steps:

1. Prepare yocto build environment
2. add meta-x32 layer, http://git.yoctoproject.org/cgit/cgit.cgi/experimental/meta-x32/
3. Add following lines in your conf/local.conf
MACHINE = "qemux86-64"
DEFAULTTUNE = "x86-64-x32"

Expected Results:

x32 image could be built out successfully

Last Result	**Not Run**
Keywords:	None

Test Case TC-1539: x32 image build boot up and check
Summary: x32 image could be built out successfully and binaries/libraries are x32 in it
Steps: 1. Prepare yocto build environment 2. add meta-x32 layer, http://git.yoctoproject.org/cgit/cgit.cgi/experimental/meta-x32/ 3. Add following lines in your conf/local.conf MACHINE = "qemux86-64" DEFAULTTUNE = "x86-64-x32" 4. build minimal image with "bitbake core-image-minimal" 5. Run the file command to know what type of elf binary is it. It should be 32bit x86-64 elf binary as seen here: $ file bin/busybox bin/busybox: setuid ELF 32-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.35, not stripped $file usr/lib/libz.so.1.2.5 usr/lib/libz.so.1.2.5: ELF 32-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, not stripped
Expected Results: x32 image could be built out successfully and binaries/libraries are x32 in it

Last Result	**Not Run**
Keywords:	None

Test Case TC-1544: Gtk+ Over DirectFB
Summary: Check if the gtk-directfb image can be built out and gtk-demo can run
Steps: 1. Download poky source and prepare the build environment . 2. Set MACHINE to qemuarm in conf/local.conf . 3. Run "bitbake core-image-gtk-directfb" to build a gtk-directfb image . 4. Bootup the gtk-directfb image and run "gtk-demo" command.
Expected Results: The gtk-directfb image can be built out and the "gtk-demo" command can run without problems.

Last Result	**Not Run**
Keywords:	None

Test Case TC-1545: bitbake-runtask	
Summary: Check if bitbake-runtask command could work.	
Steps: 1. Download poky source and prepare the build environment . 2. Run “bitbake-runtask” command to build some packages. For example, run the following commands: "bitbake-runtask man_1.6f do_fetch" "bitbake-runtask man_1.6f do_unpack" "bitbake-runtask man_1.6f do_patch" "bitbake-runtask man_1.6f do_configure" "bitbake-runtask man_1.6f do_compile" "bitbake-runtask man_1.6f do_install" "bitbake-runtask man_1.6f do_populate_lic" "bitbake-runtask man_1.6f do_populate_sysroot" 3. Check the return value of each command by using “echo $?” and check the log file in work directory.	
Expected Results: The return value of each command should be “0” and no error message in log file.	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1546: autoconf-nativesdk and automake-nativesdk support	
Summary: Check if toolchain support autoconf-nativesdk and automake-nativesdk.	
Steps: 1. Install toolchain tarball and setup cross compile environment. 2. Check if there are “autoconf” and “automake” commands in toolchain tarball. Check if there is a option “--with-libtool-sysroot” in ${CONFIGURE_FLAGS}. 3. Download iptables project. There is a macro “AM_PROG_LIBTOOL” in configure.ac. With the cross compile environment, run "autoreconf", "./configure ${CONFIGURE_FLAGS}", "make", "make install DESTDIR=/opt/tmp"	
Expected Results: The “autoconf” and “automake” commands should be contained in meta-toolchain. When running “./configure ${CONFIGURE_FLAGS}”, there is no warning message like: “WARNING: unrecognized options: --with-libtool-sysroot”	
Last Result	**Not Run**
Keywords:	None

1.9 Test Suite : BSP specific

Test Case TC-1390: syslinux for non-EFI BSPs	
Summary: check if non-EFI BSP images use syslinux for kernel loading	
Steps: 1. Get the non-EFI BSP images from autobuilder or build them on local machine 2. install the images into harddisk and boot from the harddisk 3. syslinux should be used for kernel loading	
Expected Results: syslinux should be used for kernel loading	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1391: EFI boot	
Summary: check if EFI booting is supported by Intel BSPs	
Steps: 1. Download EFI BSP images from autobuilder or build them on local machine 2. Burn the images into harddisk 3. boot from harddisk and choose EFI shell to boot from EFI 4. check system could boot up with EFI	
Expected Results: check if EFI booting is supported by Intel BSPs	
Last Result	**Not Run**
Keywords:	None

Test Case TC-1540: RTC
Summary: Check if RTC(Real Time Clock) can work correctly
Steps: 1. Read time from RTC registers. root@localhost:/root> hwclock -r Sun Mar 22 04:05:47 1970 -0.001948 seconds 2. Set system current time root@localhost:/root> date 062309452008 3. Synchronize the system current time to RTC registers

root@localhost:/root> hwclock -w

4. Read time from RTC registers

root@localhost:/root> hwclock -r

5. Reboot target and read time from RTC again.

Expected Results:
Can read and set the time successful

Last Result	**Not Run**
Keywords:	None

Test Case TC-1541: Watchdog
Summary:
Check if watchdog can reset the target system
Steps:
1. Check if watchdog device exist in /dev/ directory 2. Run command “echo 1 > /dev/watchdog” and wait for 60s. Then the target will reboot.
Expected Results:
The watchdog device exist in /dev/ directory and can reboot the target.

Last Result	**Not Run**
Keywords:	None

Test Case TC-1542: SATA
Summary:
Test general use of SATA device on target, like mount, umount, read and write.
Steps:
1. Run “fdisk” command to create partition on SATA disk. 2. Mount/Umount mke2fs /dev/sda1 mount -t ext2 /dev/sda1 /mnt/disk umount /mnt/disk

3. Read/Write (filesystem) touch /mnt/disk/test.txt echo “abcd” > /mnt/disk/test.txt cat /mnt/disk/test.txt 4. Read/Write (raw) dd if=/dev/sda1 of=/tmp/test bs=1k count=1k This command will read 1MB from /dev/sda1 to /tmp/test
Expected Results: The SATA device can mount, umount, read and write

Last Result	**Not Run**
Keywords:	None

Test Case TC-1543: I2C/EEPROM
Summary: Check if target can support EEPROM
Steps: 1. Check eeprom device exist in /sys/bus/i2c/devices/ 2. Run "hexdump eeprom" command root@mpc8315e-rdb:/sys/bus/i2c/devices/1-0051> hexdump eeprom 0000000 9210 0b02 0211 0009 0b52 0108 0c00 3c00 0000010 6978 6930 6911 208c 7003 3c3c 00f0 8381
Expected Results: Hexdump can read data from eeprom

Last Result	**Not Run**
Keywords:	None